AN INTRODUCTION TO EILLIPTIC CURVE CRYPTOGRAPHY

MAXIE D. SCHMIDT
SCHOOL OF MATHEMATICS
GEORGIA INSTITUTE OF TECHNOLOGY
ATLANTA, GA 30332 USA

1. Introduction

Since the mid 1970's public key cryptography (PKC) has historically been based on the RSA algorithm which relies on the ease with which we may multiply prime numbers and correspondingly the difficulty we find in factorizing large integers into their component primes. This reflects the historical use of a so-called *trapdoor function* which is easy to compute in one direction (i.e., the encryption process) but which is hard to invert. More recently, as computational power has increased the need for the use of larger and larger primes with RSA, and hence substantially larger key sizes, has suggested that the long term continued use of RSA as a public key cryptography standard intractable. Now enter the use of *elliptic curve cryptography* (ECC) as a cryptographic standard.

Since the mid 2000's the use of ECC to encrypt sensitive data such as classified data and online transactions has become increasingly more common over the more traditional RSA. ECC-based crypto systems typically afford users equivalent security to RSA for a significant reduction in key size which results in implementations of ECC-based encoding schemes requiring smaller chip sizes and lower power constraints in mobile computing environments and other applications. For example, one estimation cites that a 4096-bit RSA encryption scheme is equivalent to a 313-bit key ECC scheme. Additionally, another study found that the time to generate a 512-bit RSA key on a particular hardware setup took approximately 3.4 minutes, where an equivalent routine to generate a 163-bit ECC-DSA key required only 0.597 seconds [6, §6]. Table 1.1 is reproduced from the reference [3] to compare the approximate key sizes necessary for equivalent security under each of symmetric key, RSA, and ECC encryption schemes.

Symmetric Key Size	**RSA Key Size**	**ECC Key Size**
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Table 1.1. *A comparison of key sizes needed for equivalent security under several popular encryption schemes. Clearly, the lowest key sizes are afforded by symmetric-key encryption methods, but these implementations are complicated by the need for secure key exchange protocols which must be in place before any exchanges of encrypted data can take place. The equivalent key sizes between RSA and ECC-based schemes reflect that an ECC cryptosystem can be implemented with a significantly lower bar for computational resources than traditional PKC encryption systems based on RSA (or Diffie-Hellman). Thus ECC is more suitable for the setting of computationally constrained applications such as mobile environments where power consumption must be at a minimum footprint.*

Date: Monday 9th April, 2018.

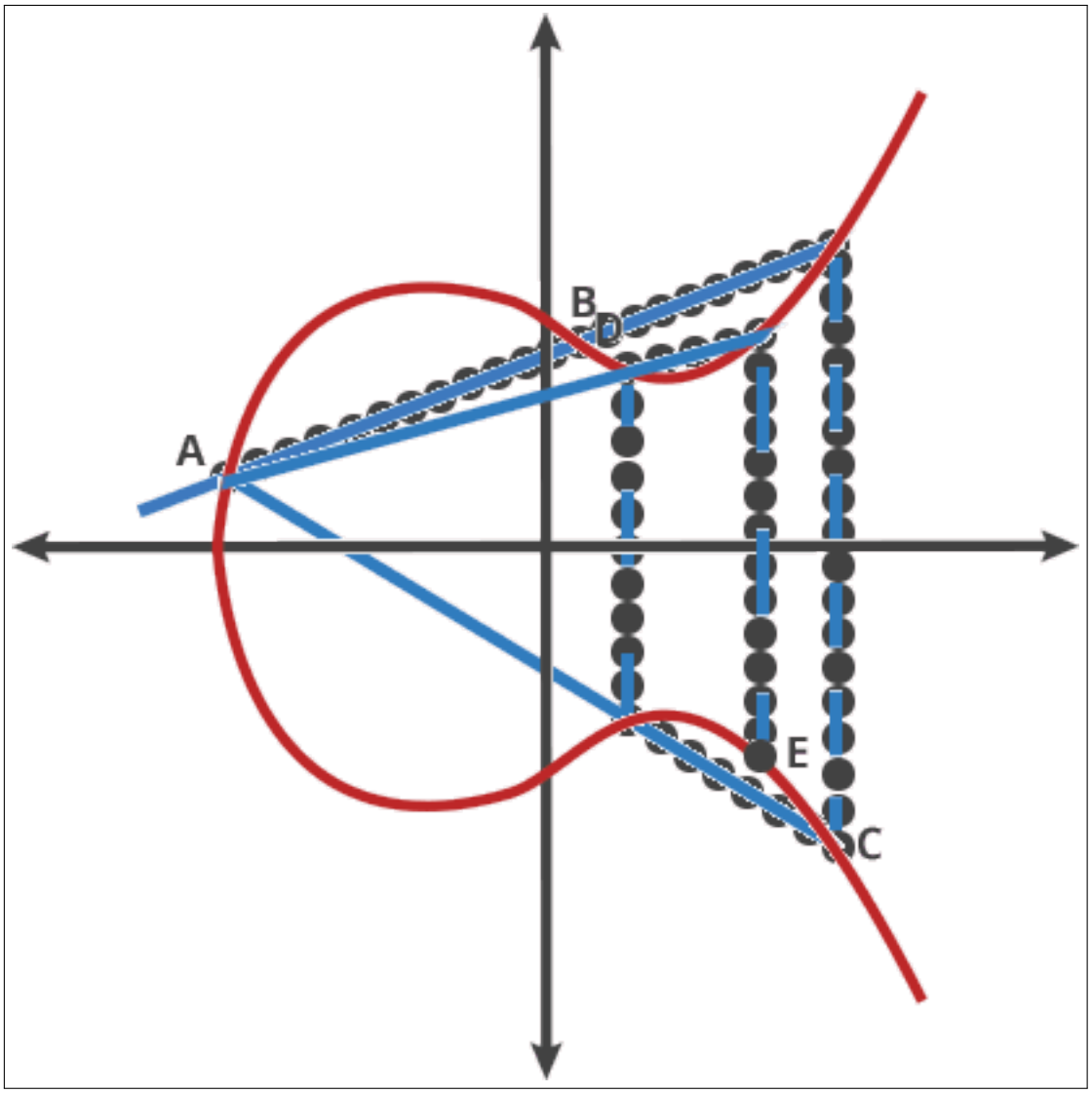

Figure 1.1. *The elliptic curve "dot" (+) operation for integer multiplication (or classically exponentiation) shown on the points A, B, C, D, E satisfying $A+B=C$, $A+A=B$, $A+B=C$, $A+C=D$, and $A+D=E$ [5].*

The trapdoor function at play in ECC schemes is based on the diagram shown in Figure 1.1 and Figure 1.2. Namely, given an elliptic curve E/F which assumes values in some, let's say, finite field F satisfying a *Weierstrass equation* of the form $y^2 = x^3 + ax + b$ for some constants a and b as

$$E(F) = \{\infty\} \cup \left\{(x, y) \in F^2 : y^2 = x^3 + ax + b\right\},$$

we see that in one direction it is easy to compute the operation $A+B=C$ and more generally the point A dotted with itself $n \geq 1$ times as in the figure:

$$C_n = \underbrace{A + A + \cdots + A}_{n \text{ times}} \longmapsto nA. \qquad \text{(Exponentiation / Integer Multiplication in } E\text{)}$$

Correspondingly, given an initial point A "dotted" with itself n times to obtain a final point C_n, the problem of finding the operation number n given only A and C_n is computationally hard. In other words, the *elliptic curve discrete logarithm problem* (ECDLP) of obtaining the discrete logarithm of a random point on E/F given a public base point on the curve is computationally infeasible. For prime p and $x, y \in \mathbb{Z}$ such that both $X, Y \neq 0 \bmod p$, the ECDLP is more precisely phrased as finding an integer

n such that $nX = Y$. In our setting, this is equivalent to the phrasing of the classical discrete logarithm problem in the form of

$$X^n \equiv Y \pmod{p}. \qquad \text{(The Classical Discrete Logarithm Problem)}$$

This is the trapdoor function scheme on which ECC is built. More precisely, the group law for addition (or *dotting* as above) is defined according to the next rules (*cf.* Figure 1.2) [6, §2.2]. The related rule defining exponentiation (or an integer times a point in additive notation), or the effect of n in the ECDLP, can also be found given explicitly in algorithmic steps for even / odd n in the same subsection of Washinton's book. Other representations of and coordinate systems for elliptic curves E which can be employed to yield computationally faster exponentiation rules are discussed in [6, §2.6].

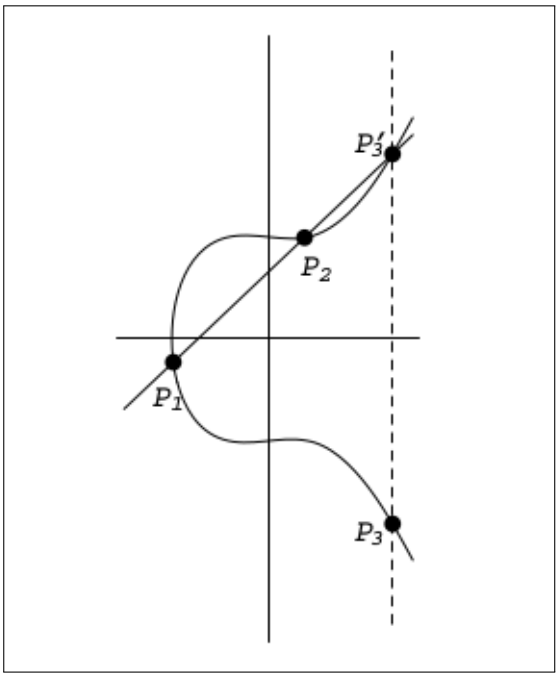

Figure 1.2. *The elliptic curve group operation (+ or ·) written additively as* $P_1 + P_2 = P_3$ [6, §2.2].

Remark 1.1 (Interpretations of the Group Law Over Finite Fields)**.** The interpretation of the algorithm for the group law given in Deinition 1.2 and demonstrated in Figure 1.1 and in Figure 1.2 can be visualized as the same procedure we perform in the cases over $\mathbb{R}$ or $\mathbb{C}$. In particular, the steps outlined above correspond to the same operation of intersecting the curve with lines and reflecting to obtain new points as in the less restrictive cases of $\mathbb{R}$ / $\mathbb{C}$. Repeatedly adding the same point to itself as in the posint of the discrete logarithm problem corresponds to "integer multiplication" in our additive notation which is effectively the action of exponentiation in the classical discrete logarithm problem stated above.

Definition 1.2 (The Group Law for Elliptic Curves)**.** Let $E : y^2 = x^3 + ax + b$ be an elliptic curve and let $P_1 := (x_1, x_2)$ and $P_2 := (x_2, y_2)$ be points on the curve with $P_1, P_2 \neq \infty$. We define the addition (or *dot* operation) $P_3 := P_1 + P_2$ for some coordinates $P_3 := (x_3, y_3)$ according to the following rules:

1. If $x_1 \neq x_2$, then $(x_3, y_3) = (m^2 - x_1 - x_2, m(x_1 - x_3) - y_1)$ where $m = (y_2 - y_1)/(x_2 - x_1)$;
2. If $x_1 = x_2$ but $y_1 \neq y_2$, then $P_3 = \infty$;
3. If $P_1 = P_2$ and $y_1 \neq 0$, then $(x_3, y_3) = (m^2 - 2x_1, m(x_1 - x_3) - y_1)$ where $m = \frac{1}{2} \cdot (3x_1^2 + a)/y_1$;
4. If $P_1 = P_2$ and $y_1 = 0$, then $P_3 = \infty$.

Note that for all points P on E, we define the special case of the group operation dotted with ∞ to be $P \cdot \infty = P$ (i.e., ∞ is the additive identity). The elliptic curve $E(F)$ defined such that $a, b \in F$ for a field F is closed under the group operation $+$. Moreover, the group operation in $E(F)$ is commutative, associative, and invertible which implies that the points on E/F are an additive abelian group with identity.

In general, the fastest known algorithms for finding a solution to the ECDLP run in $O(\sqrt{n})$, e.g., the *baby step, giant step* method, *Pollard's* ρ method, or the *Pohlig-Hellman* method [6, §4.3.4; §5.2]. Thus in order to obtain the equivalent of 2^k-bit RSA encryption, one effectively needs to compute points of an elliptic curve E over finite field $\mathbb{F}_q$ of size at least $q \approx 2^{2^{k+1}}$. The technology standards governing body NIST has published 15 recommended elliptic curves for use in ECC encryption schemes over prime fields $\mathbb{F}_p$ for several specific p and over $\mathbb{F}_{2^m}$ for several particular m (*cf.* [4, 1]). The NIST recommended standard then includes an overall total of five prime curves and ten binary curves. We will consider the computational algorithms and coding task of implementing an ECC encryption scheme over an arbitrary elliptic curve taken from the tables in [2] over a suitably chosen finite field given our desired key sizes in the application in the next section.

2. Elliptic curve cryptosystems in practice

2.1. **A theoretical public key cryptosystem implementation using ECC.** We describe the general setup and algorithm for an *El-Gamal*-based cryptosystem built for public key encryption (following as in [6, §6.4])[1]. Suppose that we fix the parameters (E, b, q) in an ECC-based cryptosystem which involves the computation of points over $E(\mathbb{F}_q)$ with a key size of b-bits and a corresponding suitable choice of $q \geq 2^{2b}$ for RSA-equivalent security. The general setup for a data protocol is as follows:

> **Scheme for an El-Gamal ECC PK Cryptosystem:**
> *The participant Alice (A) wants to send messages to another participant Bob (B) using a PKC encryption method. Bob chooses his favorite elliptic curve $E/\mathbb{F}_q$ for an appropriate finite field $\mathbb{F}_q$ such that the ECDLP is sufficiently hard over $E(\mathbb{F}_q)$ (see the above considerations). In addition, Bob selects a point P on the curve with order of some large prime as a starting point. Bob picks a secret integer parameter s and computes the point $B := sP$ over $E(\mathbb{F}_q)$. This is all to say that Bob's public key consists of the tuple $P_{\text{pub}} = (E, q, P, B)$ and his private key is the non-broadcast $P_{\text{priv}} = (s)$. As in RSA, Bob publishes his selection of P_{pub} in a public repository accessible to Alice and others and keeps his private key P_{priv} as a secret known only to him for decryption purposes on his backend. There is an adversarial evesdropper Eve (E) on the communication channel who seeks to spy on Alice and Bob's presumed secret message exchange with this PKC protocol. She has access to Bob's public key P_{pub} and all cyphertext information sent by Alice to Bob.*

The following describes an algorithmic procedure by which Alice can send encrypted messages (presumably securely) to Bob under this ECC-based PKC scheme:

Algorithm 2.1 (Secure Message Exchange Protocol)**.** To perform the exchange of the sensitive plaintext message M from A to B, we use the following steps to implement our exchange protocol:

1. A obtains B's public key data $P_{\text{pub}} = (E, q, P, B)$ from the public repository where it is stored;
2. A encodes her plaintext message as a point M in $E(\mathbb{F}_q)$ using the procedure outlined in Section 2.2.
3. A choses a new distinct random $k \in \mathbb{Z} \setminus \{0\}$ and computes the cyphertext $M_1 := kP$;
4. Alice completes the associated cyphertext pair by encrypting $M_2 := M + kB$;
5. A sends (M_1, M_2) over the communication channel to B;
6. B decrypts A's cyphertext transmission into cleartext according to the formula $M = M_2 - sM_1$.

Correctness of the Decryption Scheme by Bob. We now prove that provided Bob receives both of Alice's intended messages M_1, M_2 intact, i.e., that there has been no corruption nor malicious tampering with the exchanged data, that Bob decrypts his received cyphertext pair into the correct original message M sent by Alice. In particular, we suppose that given (M_1, M_2) (in assumed pristine condition from the volatile communication channel) Bob decrypts the cypertext into the message

$$\widetilde{M} := M_2 - s \cdot M_1. \qquad \text{(Bob's Decryption Formula)}$$

[1] We use this example of a cryptosystem based on ECC as a relatively straightforward to understand mock toy example for the purposes of exposition and education. It is pointed out in the reference [6, p. 175] that this particular ECC-based scheme providing an implementation of a PKC cryptosystem is not widely used in practice.

Now we must show by direct calculation that $\widetilde{M} = M$:

$$\begin{aligned}\widetilde{M} &= M_2 - s \cdot M_1 \\ &= (M + k \cdot B) - s \cdot (kP) \\ &= M + k(sP) - skP \\ &= M. \qquad \text{(Commutativity of the Group Operation on E)}\end{aligned}$$

Thus Bob decrypts the cyphertext sent by Alice into the correct message M regardless of the selection of the protocol parameters (s, k). □

Remarks on the Data Exchange Scheme. There are two primary remarks to be made on the PKC-based secret message exchange protocol defined (and subsequently proved correct) above. We remark on the following underlying aspects of our scheme:
(i) The Effectiveness of Attacks by Eve:
Without additional data, or faults in inadvertantly exposing additional secure data in the exchange by participants (see next point), Eve is more or less limited in her ability to spy on the encrypted communication by the difficulty of solving the ECDLP over sufficiently large finite fields. Suppose on the other hand that Eve possesses the ability to solve the ECDLP by black box. Then Eve can easily decrypt the original message sent by Alice given her knowledge of $(P_{\text{pub}}; M_1, M_2)$ as follows: either (a) use the DLP oracle with P and B to find s and then compute the decrypted message directly as in Bob's formula; or (b) obtain k using P and M_1 and then compute the resulting $M = M_2 - kB$. In the abscence of a reliable and fast solution to the ECDLP, Eve's ability to evesdrop on the communication channel *should* be otherwise significantly restricted.
(ii) The Randomness of Alice's Cyphertext Generation Method:
It is imperative that Alice use a fresh (distinct) randomized encryption parameter k for each message she sends to Bob to prevent the eavesdropper Eve from employing her knowledge of the conversation history between Alice and Bob to obtain "easy" correct decryptions of future messages exchanged by the two with the same key pair and parameters. Consider the scenario where Alice uses that same integer k to encrypt both of the plaintext messages M and M' into cyphertext which she then sends to Bob over the channel where Eve reigns as supreme Big Sister. Eve is able to distinguish this multiple use of the parameter k because she can detect that $M_1 = M_1'$ in the protocol. We then suppose further that Eve obtains the plaintext for the first message M which is purportedly leaked as a sales announcement by the company where Alice and Bob are employees. Since Eve is knowledgable and opportunistic in exploiting such leaked, or otherwise compromised sensitive data, she computes the message difference

$$M_2' - M_2 = M' - M,$$

and then observes that the calculation of

$$M' = M - M_2 + M_2',$$

decodes to a juicy plaintext message. In other words, Alice's mistake in duplicating the encryption parameter k over multiple messages exchanged with Bob together with the leaked sensitive data has allowed Eve to exploit this situation and obtain the decrypted second message while managing to avert any computational challenge the hardness of the ECDLP implies to offer as protection for the communication. □

2.2. **A scheme for encoding plaintext data as points over $E(\mathbb{F}_q)$.** The only substep remaining in the El-Gamal PKC cryptosystem based on ECC in the previous subsection is a reasonable implementation of an encoding scheme for messages over $E(\mathbb{F}_q)$. Washington's book describes a method due to Koblitz for representing (let's say) ascii-encoded integers for some plaintext message as a point on the explicitly chosen elliptic curve E over some finite field $\mathbb{F}_q$ in the special case where $q \equiv p$ is prime [6, §6.3]. Proceeding as in the suggestion proposed there, Koblitz's method can be generalized to form an encoding scheme to points on E over an *arbitrary* finite field case of $\mathbb{F}_q$. We next present the details of the more general cases handled by such an encoding system in this subsection below as an extension of Washington's text for this project.

We suppose that (E, b, q) are fixed parameters known as in the PKC protocol described in Section 2.1. We also require an efficient black box oracle for computing

$$\text{FFSQ}_q\left(s^2 \bmod q\right) \mapsto s \pmod q, \qquad \text{(Square Root Operation Over Finite Fields)}$$

effectively providing an integer-factorization-based implementation of finding the square root s of an integer $s^2 \in \mathbb{F}_q$. This cannot necessarily be done in sub-exponential running time depending on the s, q. However, in the special case where q is prime, this operation can be done in polynomial time with respect to q by the *Tonelli-Shanks algorithm.*

Algorithm 2.2 (Message Encoding Procedure)**.** Our procedure for encoding any integer $0 \leq m < q/M$ into a point (x_m, y_m) on $E(\mathbb{F}_q)$ is given by the following steps:

1. For $1 \leq j < M$, repeat the next three substeps steps until a suitable encoding $(x_j, y_j) \mapsto (x_m, y_m)$ is found:
2. Define: $x_j := M \cdot m + j$;
3. Compute: $s_j := x_j^3 + ax_j + b$;
4. If s_j is a square modulo q, then set $y_j := \text{FFSQ}_q(s_j)$ and return the pair $(x_j, y_j) \mapsto (x_m, y_m)$ as our encoding.
5. Otherwise, repeat the process for $j \mapsto j + 1$.

Note that the encoding parameter M must be suitably defined for each q to ensure that we can find an encoding for each message m with high probability. In particular, we expect that the s_j computed in step 2 above is essentially a random element of $\mathbb{F}_q^{\times}$, so that with probability approximately $\left(\frac{q-1}{2q}\right)^M$ we obtain a valid encoding after the maximum of M tries in our algorithm.

Given an encoding $(M; x_m, y_m)$ of m produced by the above algorithm, the associated reversal, or decoding procedure is performed as follows:

$$m = \left\lfloor \frac{x_m}{M} \right\rfloor. \qquad \text{(Message Decoding Procedure)}$$

2.3. **The implementation of baby-step, giant-step in our accompanying program.** This section describes an algorithmic implementation of the *baby-step, giant-step* method for solving the ECDLP employed in the accompanying demo program which demonstrates the running time necessary for an evesdropper (Eve) to naively compute the solution to the DLP over a finite field in approximately $O(\sqrt{q})$ time and $O(\sqrt{q})$. Our method follows along the same lines as the procedure from [6, §5.2.1; *cf.* §4.3.4].

Suppose that we are given points $P, Q \in E(\mathbb{F}_q)$ and we seek to find an integer s such that $P^s = Q \pmod q$ (written multiplicatively), or more commonly $sP = Q$ in the additive notation employed below, where a priori we know that such an integer does in fact exist as Q has been computed using our cryptosystem from Section 2.1. Then an algorithm known as D. Shank's baby-step, giant-step method provides an inversion of this version of the ECDLP in our ECC-based PKC context from above.

Algorithm 2.3 (Baby-Step, Giant-Step)**.** Perform the following steps as the implementation of our baby-step, giant-step procedure for computing the discrete logarithm over $E(\mathbb{F}_q)$:

1. Fix any integer $m \geq \lceil \sqrt{q} \rceil$ and compute the corresponding exponent mP;
2. Compute and store an array of the values iP for each $0 \leq i < m$ (**baby step**);
3. For $0 \leq j < m$, compute the points $D_j := Q - jmP$ (**giant step**). If D_j is in the array of stored values from the previous step, continue to the next substep.
4. If $iP = D_j$, then we must have that $Q = sP$ with $s \equiv i + jm \bmod q$.

Correctness of the Baby-Step, Giant-Step Algorithm. We first observe that $m^2 \geq q$ so that the solution s to the ECDLP is in the range $0 \leq s < m^2$. Next, we can express the solution in the form of $s = s_0 + ms_1$ for some reduced $s_0 \equiv s \bmod q$ in the range $0 \leq s_0 < m$ and $s_1 := (s - s_0)/m \in \mathbb{Z}$ defined such that $0 \leq s_1 < m$. In the algorithmic procedure we just defined, if $i = s_0$ and $j = s_1$, then we compute that

$$Q - s_1 mP = sP - s_1 mP = s_0 P.$$

Hence, the algorithm terminates with a solution. □

3. Conclusions

Various weaknesses in ECC encryption related to attacks which make computing solutions to the ECDLP easier are discussed in [6, §5.3–5.5]. Other attacks on ECC cryptographic schemes include timing-based side-channel attacks and hypothetical (though currently infeasible) attacks possible given the existence of a quantum computer with an implementation of *Shor's algorithm*. The first type of attack can be mitigated by the use of *Edwards curves* which are a special family of elliptic curves which are not succeptible to such doubling operations on elliptic curves. The latter would be a significant disadvantage to ECC-based cryptographic routines as given the plausible advent of quantum computing capabilities in practice ECC schemes require considerably less computational resources to break than a corresponding equivalent RSA encryption scheme. As with everything else, there are fears by some crypto-experts asserting that large government agencies like the NSA have inserted so-called asymmetric *kleptograhic backdoors* into existing ECC schemes whereby the resulting cyphertext obtained through these schemes is much more easily reverse engineered into plaintext by the agency in command of the inserted backdoor parameters, or attacker's known private key into the cryptosystem. Moreover, the cyphertext outputted from a kleptographically infected cryptosystem is in theory indistingishible by computational methods to an unaltered encryption scheme. Overall, ECC-based schemes are suitable for many applications including in key exchange and agreement protocols, encryption of sensitive plaintext data, digital signatures, (pseudo) random number generation, and integer factorization algorithms among others which makes them a computationally lightweight viable alternative to their PKC predecessors in modern cryptograhic applications.

References

[1] D. R. L. Brown, SEC 2: Recommended Elliptic Curve Domain Parameters, *Certicom Research*, http://www.secg.org/sec2-v2.pdf (2010).
[2] J. E. Cremona, Elliptic Curve Data, http://johncremona.github.io/ecdata/ (2017).
[3] J. Garcia, Elliptic Curve Crypto, The Basics, *Hacker Noon Online* (2017). Link available here.
[4] R. G. Kammer, Digital Signature Standard (DSS), *U.S. Department of Commerce / NIST* (2000). Link available here.
[5] N. Sullivan, A (relatively easy to understand) primer on elliptic curve cryptography, *Ars Technica Online* (2013). Link available here.
[6] L. C. Washington, *Elliptic Cruves: Number Theory and Cryptography* (Section Edition), Chapman and Hall / CRC, 2008.